Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions

Gediminas Adomavicius[1] and Alexander Tuzhilin[2]

Abstract–The paper presents an overview of the field of recommender systems and describes the current generation of recommendation methods that are usually classified into the following three main categories: content-based, collaborative, and hybrid recommendation approaches. The paper also describes various limitations of current recommendation methods and discusses possible extensions that can improve recommendation capabilities and make recommender systems applicable to an even broader range of applications. These extensions include, among others, improvement of understanding of users and items, incorporation of the contextual information into the recommendation process, support for multi-criteria ratings, and provision of more flexible and less intrusive types of recommendations.

Index Terms–Recommender systems, collaborative filtering, rating estimation methods, extensions to recommender systems.

1. Introduction

Recommender systems became an important research area since the appearance of the first papers on collaborative filtering since the mid-1990s [45, 86, 97]. There has been much work done both in the industry and academia on developing new approaches to recommender systems over the last decade. The interest in this area still remains high because it constitutes a problem-rich research area and because of the abundance of practical applications that help users to deal with information overload and provide personalized recommendations, content and services to them. Examples of such applications include recommending books, CDs and other products at Amazon.com [61], movies by MovieLens [67], and news at VERSIFI Technologies (formerly AdaptiveInfo.com) [14]. Moreover, some of the vendors have incorporated recommendation capabilities into their commerce servers [78].

However, despite all these advances, the current generation of recommender systems still requires further improvements to make recommendation methods more effective and applicable to an even broader range of real-life applications, including recommending vacations, certain

[1] G. Adomavicius is with the Carlson School of Management, University of Minnesota, 321 19th Avenue South, Minneapolis, MN 55455. Email: gedas@umn.edu.
[2] A. Tuzhilin is with the Stern School of Business, New York University, 44 West 4th Street, New York, NY 10012. Email: atuzhili@stern.nyu.edu.

types of financial services to investors, and products to purchase in a store made by a "smart" shopping cart [106]. These improvements include better methods for representing user behavior and the information about the items to be recommended, more advanced recommendation modeling methods, incorporation of various contextual information into the recommendation process, utilization of multi-criteria ratings, development of less intrusive and more flexible recommendation methods that also rely on the measures that more effectively determine performance of recommender systems.

In this paper, we describe various ways to extend capabilities of recommender systems. However, before doing this, we first present a comprehensive survey of the state-of-the-art in recommender systems in Section 2. Then we identify various limitations of the current generation of recommendation methods and discuss some initial approaches to extending their capabilities in Section 3.

2. The Survey of Recommender Systems

Although the roots of recommender systems can be traced back to the extensive work in the cognitive science [87], approximation theory [81], information retrieval [89], forecasting theories [6], and also have links to management science [71], and also to the consumer choice modeling in marketing [60], recommender systems emerged as an independent research area in the mid-1990's when researchers started focusing on recommendation problems that *explicitly* rely on the ratings structure. In its most common formulation, the recommendation problem is reduced to the problem of estimating *ratings* for the items that have not been seen by a user. Intuitively, this estimation is usually based on the ratings given by this user to other items and on some other information that will be formally described below. Once we can estimate ratings for the yet unrated items, we can recommend to the user the item(s) with the highest estimated rating(s).

More formally, the recommendation problem can be formulated as follows. Let C be the

set of all users and let *S* be the set of all possible items that can be recommended, such as books, movies, or restaurants. The space *S* of possible items can be very large, ranging in hundreds of thousands or even millions of items in some applications, such as recommending books or CDs. Similarly, the user space can also be very large – millions in some cases. Let *u* be a utility function that measures usefulness of item *s* to user *c*, i.e., $u : C \times S \to R$, where *R* is a totally ordered set (e.g., non-negative integers or real numbers within a certain range). Then for each user $c \in C$, we want to choose such item $s' \in S$ that maximizes the user's utility. More formally:

$$\forall c \in C, \quad s'_c = \arg\max_{s \in S} u(c, s) \tag{1}$$

In recommender systems the utility of an item is usually represented by a *rating*, which indicates how a particular user liked a particular item, e.g., John Doe gave the movie "Harry Potter" the rating of 7 (out of 10). However, as indicated earlier, in general utility can be an arbitrary function, including a profit function. Depending on the application, utility *u* can either be specified by the user, as is often done for the user-defined ratings, or is computed by the application, as can be the case for a profit-based utility function.

Each element of the user space *C* can be defined with a *profile* that includes various user characteristics, such as age, gender, income, marital status, etc. In the simplest case, the profile can contain only a single (unique) element, such as User ID. Similarly, each element of the item space *S* is defined with a set of characteristics. For example, in a movie recommendation application, where *S* is a collection of movies, each movie can be represented not only by its ID, but also by its title, genre, director, year of release, leading actors, etc.

The central problem of recommender systems lies in that utility *u* is usually not defined on the whole $C \times S$ space, but only on some subset of it. This means *u* needs to be *extrapolated* to the whole space $C \times S$. In recommender systems, utility is typically represented by ratings

and is initially defined only on the items previously rated by the users. For example, in a movie recommendation application (such as the one at MovieLens.org), users initially rate some subset of movies that they have already seen. An example of a user-item rating matrix for a movie recommendation application is presented in Table 1, where ratings are specified on the scale of 1 to 5. The "∅" symbol for some of the ratings in Table 1 means that the users have not rated the corresponding movies. Therefore, the recommendation engine should be able to estimate (predict) the ratings of the non-rated movie/user combinations and issue appropriate recommendations based on these predictions.

	K-PAX	Life of Brian	Memento	Notorious
Alice	4	3	2	4
Bob	∅	4	5	5
Cindy	2	2	4	∅
David	3	∅	5	2

Table 1. A fragment of a rating matrix for a movie recommender system.

Extrapolations from known to unknown ratings are usually done by (a) specifying *heuristics* that define the utility function and empirically validating its performance, and (b) *estimating* the utility function that optimizes certain performance criterion, such as the mean square error.

Once the unknown ratings are estimated, actual recommendations of an item to a user are made by selecting the highest rating among all the estimated ratings for that user, according to formula (1). Alternatively, we can recommend *N* best items to a user or a set of users to an item.

The new ratings of the not-yet-rated items can be estimated in many different ways using the methods from machine learning, approximation theory and various heuristics. Recommender systems are usually classified according to their approach to rating estimation, and in the next section, we will present such a classification that was proposed in the literature and will provide a survey of different types of recommender systems. The commonly accepted formulation of the recommendation problem was first stated in [45, 86, 97] and this problem has been studied

extensively since then. Moreover, recommender systems are usually classified into the following categories, based on how recommendations are made [8]:

- *Content-based recommendations:* the user is recommended items similar to the ones the user preferred in the past;
- *Collaborative recommendations:* the user is recommended items that people with similar tastes and preferences liked in the past;
- *Hybrid approaches:* these methods combine collaborative and content-based methods.

In addition to recommender systems that predict the *absolute* values of ratings that individual users would give to the yet unseen items (as discussed above), there has been work done on *preference-based filtering*, i.e., predicting the *relative* preferences of users [22, 35, 51, 52]. For example, in a movie recommendation application preference-based filtering techniques would focus on predicting the correct relative *order* of the movies, rather than their individual ratings. However, this paper focuses primarily on the *rating-based* recommenders, since it constitutes the most popular approach to recommender systems.

2.1 Content-based Methods

In content-based recommendation methods, the utility $u(c,s)$ of item s for user c is estimated based on the utilities $u(c,s_i)$ assigned by user c to items $s_i \in S$ that are "similar" to item s. For example, in a movie recommendation application, in order to recommend movies to user c, the content-based recommender system tries to understand the commonalities among the movies user c has rated highly in the past (specific actors, directors, genres, subject matter, etc.). Then, only the movies that have a high degree of similarity to whatever user's preferences are would get recommended.

The content-based approach to recommendation has its roots in information retrieval [7, 89] and information filtering [10] research. Because of the significant and early advancements made by the information retrieval and filtering communities and because of the importance of

several text-based applications, many current content-based systems focus on recommending items containing textual information, such as documents, Web sites (URLs), and Usenet news messages. The improvement over the traditional information retrieval approaches comes from the use of user *profiles* that contain information about users' tastes, preferences, and needs. The profiling information can be elicited from users explicitly, e.g., through questionnaires, or implicitly – learned from their transactional behavior over time.

More formally, let *Content*(*s*) be an *item profile*, i.e., a set of attributes characterizing item *s*. It is usually computed by extracting a set of features from item *s* (its content) and is used to determine appropriateness of the item for recommendation purposes. Since, as mentioned earlier, content-based systems are designed mostly to recommend text-based items, the content in these systems is usually described with *keywords*. For example, a content-based component of the Fab system [8], which recommends Web pages to users, represents Web page content with the 100 most important words. Similarly, the Syskill & Webert system [77] represents documents with the 128 most informative words. The "importance" (or "informativeness") of word k_i in document d_j is determined with some *weighting* measure w_{ij} that can be defined in several different ways.

One of the best-known measures for specifying keyword weights in Information Retrieval is the *term frequency/inverse document frequency (TF-IDF)* measure [89] that is defined as follows. Assume that N is the total number of documents that can be recommended to users and that keyword k_i appears in n_i of them. Moreover, assume that $f_{i,j}$ is the number of times keyword k_i appears in document d_j. Then $TF_{i,j}$, the term frequency (or normalized frequency) of keyword k_i in document d_j, is defined as

$$TF_{i,j} = \frac{f_{i,j}}{\max_z f_{z,j}} \quad (2)$$

where the maximum is computed over the frequencies $f_{z,j}$ of all keywords k_z that appear in the document d_j. However, keywords that appear in many documents are not useful in distinguishing between a relevant document and a non-relevant one. Therefore, the measure of inverse document frequency (IDF_i) is often used in combination with simple term frequency ($TF_{i,j}$). The inverse document frequency for keyword k_i is usually defined as

$$IDF_i = \log \frac{N}{n_i} \quad (3)$$

Then the TF-IDF weight for keyword k_i in document d_j is defined as

$$w_{i,j} = TF_{i,j} \times IDF_i \quad (4)$$

and the content of document d_j is defined as $Content(d_j) = (w_{1j}, \ldots w_{kj})$.

As stated earlier, content-based systems recommend items similar to those that a user liked in the past [56, 69, 77]. In particular, various candidate items are compared with items previously rated by the user, and the best-matching item(s) are recommended. More formally, let *ContentBasedProfile*(c) be the profile of user c containing tastes and preferences of this user. These profiles are obtained by analyzing the content of the items previously seen and rated by the user and are usually constructed using keyword analysis techniques from information retrieval. For example, *ContentBasedProfile*(c) can be defined as a vector of weights (w_{c1}, …,w_{ck}), where each weight w_{ci} denotes the importance of keyword k_i to user c and can be computed from individually rated content vectors using a variety of techniques. For example, some averaging approach, such as Rocchio algorithm [85], can be used to compute *ContentBasedProfile*(c) as an "average" vector from an individual content vectors [8, 56]. On the other hand, [77] use a Bayesian classifier in order to estimate the probability that a document

is liked. The Winnow algorithm [62] has also been shown to work well for this purpose, especially in the situations where there are many possible features [76].

In content-based systems, the utility function $u(c, s)$ is usually defined as:

$$u(c,s) = score(ContentBasedProfile(c), Content(s)) \quad (5)$$

Using the above-mentioned information retrieval-based paradigm of recommending Web pages, Web site URLs, or Usenet news messages, both *ContentBasedProfile*(c) of user c and *Content*(s) of document s can be represented as TF-IDF vectors $\vec{w}_c$ and $\vec{w}_s$ of keyword weights. Moreover, utility function $u(c, s)$ is usually represented in information retrieval literature by some scoring heuristic defined in terms of vectors $\vec{w}_c$ and $\vec{w}_s$, such as cosine similarity measure [7, 89]:

$$u(c,s) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c \cdot \vec{w}_s}{\| \vec{w}_c \|_2 \times \| \vec{w}_s \|_2} = \frac{\sum_{i=1}^{K} w_{i,c} w_{i,s}}{\sqrt{\sum_{i=1}^{K} w_{i,c}^2} \sqrt{\sum_{i=1}^{K} w_{i,s}^2}} \quad (6)$$

where K is the total number of keywords in the system.

For example, if user c reads many online articles on the topic of bioinformatics, then content-based recommendation techniques will be able to recommend other bioinformatics articles to user c. This is the case, because these articles will have more bioinformatics-related terms (e.g., "genome", "sequencing", "proteomics") than articles on other topics, and, therefore, *ContentBasedProfile*(c), as defined by vector $\vec{w}_c$, will represent such terms k_i with high weights w_{ic}. Consequently, a recommender system using the cosine or a related similarity measure will assign higher utility $u(c, s)$ to those articles s that have high-weighted bioinformatics terms in $\vec{w}_s$ and lower utility to the ones where bioinformatics terms are weighted less.

Besides the traditional heuristics that are based mostly on information retrieval methods, other techniques for content-based recommendation have also been used, such as Bayesian classifiers [70, 77] and various machine learning techniques, including clustering, decision trees,

and artificial neural networks [77]. These techniques differ from information retrieval-based approaches in that they calculate utility predictions based not on a heuristic formula, such as a cosine similarity measure, but rather are based on a *model* learned from the underlying data using statistical learning and machine learning techniques. For example, based on a set of Web pages that were rated as "relevant" or "irrelevant" by the user, [77] use the naïve Bayesian classifier [31] to classify unrated Web pages. More specifically, the naïve Bayesian classifier is used to estimate the following probability that page p_j belongs to a certain class C_i (e.g., relevant or irrelevant) given the set of keywords $k_{1,j}$, …, $k_{n,j}$ on that page:

$$P(C_i \mid k_{1,j} \& \ldots \& k_{n,j}) \tag{7}$$

Moreover, [77] use the assumption that keywords are independent and, therefore, the above probability is proportional to

$$P(C_i)\prod_x P(k_{x,j} \mid C_i) \tag{8}$$

While the keyword independence assumption does not necessarily apply in many applications, experimental results demonstrate that naïve Bayesian classifiers still produce high classification accuracy [77]. Furthermore, both $P(k_{x,j} \mid C_i)$ and $P(C_i)$ can be estimated from the underlying training data. Therefore, for each page p_j, the probability $P(C_i \mid k_{1,j} \& \ldots \& k_{n,j})$ is computed for each class C_i, and page p_j is assigned to class C_i having the highest probability [77].

While not explicitly dealing with providing recommendations, the *text retrieval* community has contributed several techniques that are being used in content-based recommender systems. One example of such technique would be the research on *adaptive filtering* [101, 112], which focuses on becoming more accurate at identifying relevant documents incrementally, by observing the documents one-by-one in a continuous document stream. Another example would be the work on *threshold setting* [84, 111], which focuses on determining the extent to which

documents should match a given query in order to be relevant to the user. Other text retrieval methods are described in [50] and can also be found in the proceedings of the Text Retrieval Conference (TREC) (http://trec.nist.gov).

As was observed in [8, 97], content-based recommender systems have several limitations that are described in the rest of this section.

Limited content analysis. Content-based techniques are limited by the features that are explicitly associated with the objects that these systems recommend. Therefore, in order to have a sufficient set of features, the content must either be in a form that can be parsed automatically by a computer (e.g., text), or the features should be assigned to items manually. While information retrieval techniques work well in extracting features from text documents, some other domains have an inherent problem with automatic feature extraction. For example, automatic feature extraction methods are much harder to apply to the multimedia data, e.g., graphical images, audio and video streams. Moreover, it is often not practical to assign attributes by hand due to limitations of resources [97].

Another problem with limited content analysis is that, if two different items are represented by the same set of features, they are indistinguishable. Therefore, since text-based documents are usually represented by their most important keywords, content-based systems cannot distinguish between a well-written article and a badly written one, if they happen to use the same terms [97].

Over-specialization. When the system can *only* recommend items that score highly against a user's profile, the user is limited to being recommended items similar to those already rated. For example, a person with no experience with Greek cuisine would never receive a recommendation for even the greatest Greek restaurant in town. This problem, which has also been studied in other domains, is often addressed by introducing some randomness. For example, the use of

genetic algorithms has been proposed as a possible solution in the context of information filtering [98]. In addition, the problem with over-specialization is not only that the content-based systems cannot recommend items that are different from anything the user has seen before. In certain cases, items should not be recommended if they are *too similar* to something the user has already seen, such as different news article describing the same event. Therefore, some content-based recommender systems, such as DailyLearner [13], filter out items not only if they are too different from user's preferences, but also if they are too similar to something the user has seen before. Furthermore, [112] provide a set of five redundancy measures to evaluate whether a document that is deemed to be relevant contains some novel information as well. In summary, the *diversity* of recommendations is often a desirable feature in recommender systems. Ideally, the user should be presented with a *range* of options and not with a homogeneous set of alternatives. For example, it is not necessarily a good idea to recommend all movies by Woody Allen to a user who liked one of them.

New user problem. The user has to rate a sufficient number of items before a content-based recommender system can really understand user's preferences and present the user with reliable recommendations. Therefore, a new user, having very few ratings, would not be able to get accurate recommendations.

2.2 Collaborative Methods

Unlike content-based recommendation methods, *collaborative* recommender systems (or *collaborative filtering systems*) try to predict the utility of items for a particular user based on the items previously rated by *other users*. More formally, the utility $u(c, s)$ of item s for user c is estimated based on the utilities $u(c_j, s)$ assigned to item s by those users $c_j \in C$ who are "similar" to user c. For example, in a movie recommendation application, in order to recommend movies to user c, the collaborative recommender system tries to find the "peers" of user c, i.e., other

users that have similar tastes in movies (rate the same movies similarly). Then, only the movies that are most liked by the "peers" of user *c* would get recommended.

There have been many collaborative systems developed in the academia and the industry. It can be argued that the Grundy system [87] was the first recommender system, which proposed to use *stereotypes* as a mechanism for building models of users based on a limited amount of information on each individual user. Using stereotypes, the Grundy system would build individual user models and use them to recommend relevant books to each user. Later on, the Tapestry system relied on each user to identify like-minded users manually [38]. GroupLens [53, 86], Video Recommender [45], and Ringo [97] were the first systems to use collaborative filtering algorithms to *automate* prediction. Other examples of collaborative recommender systems include the book recommendation system from Amazon.com, the PHOAKS system that helps people find relevant information on the WWW [103], and the Jester system that recommends jokes [39].

According to [15], algorithms for collaborative recommendations can be grouped into two general classes: *memory-based* (or *heuristic-based*) and *model-based*.

Memory-based algorithms [15, 27, 72, 86, 97] essentially are heuristics that make rating predictions based on the entire collection of previously rated items by the users. That is, the value of the unknown rating $r_{c,s}$ for user *c* and item *s* is usually computed as an aggregate of the ratings of some other (usually the *N* most similar) users for the same item *s*:

$$r_{c,s} = \underset{c' \in \hat{C}}{\text{aggr}}\, r_{c',s} \tag{9}$$

where $\hat{C}$ denotes the set of *N* users that are the most similar to user *c* and who have rated item *s* (*N* can range anywhere from 1 to the number of all users). Some examples of the aggregation function are:

(a) $r_{c,s} = \frac{1}{N}\sum_{c' \in \hat{C}} r_{c',s}$ (b) $r_{c,s} = k\sum_{c' \in \hat{C}} sim(c,c') \times r_{c',s}$ (c) $r_{c,s} = \overline{r}_c + k\sum_{c' \in \hat{C}} sim(c,c') \times (r_{c',s} - \overline{r}_{c'})$ (10)

where multiplier k serves as a normalizing factor and is usually selected as $k = 1 \Big/ \sum_{c' \in \hat{C}} |\, sim(c,c') \,|$, and where the average rating of user c, $\overline{r}_c$, in (10c) is defined as

$$\overline{r}_c = \left(1/|\, S_c \,|\right) \sum_{s \in S_c} r_{c,s} \text{ , where } S_c = \{ s \in S \mid r_{c,s} \neq \varnothing \}^{3}. \quad (11)$$

In the simplest case, the aggregation can be a simple average, as defined by expression (10a). However, the most common aggregation approach is to use the weighted sum, shown in (10b). The similarity measure between the users *c* and *c'*, *sim*(*c, c'*), is essentially a distance measure and is used as a weight, i.e., the more similar users *c* and *c'* are, the more weight rating $r_{c',s}$ will carry in the prediction of $r_{c,s}$. Note that *sim*(*x,y*) is a heuristic artifact that is introduced in order to be able to differentiate between levels of user similarity (i.e., to be able to find a set of "closest peers" or "nearest neighbors" for each user) and at the same time simplify the rating estimation procedure. As shown in (10b), different recommendation applications can use their own user similarity measure, as long as the calculations are normalized using the normalizing factor *k*, as shown above. The two most commonly used similarity measures will be described below. One problem with using the weighted sum, as in (10b), is that it does not take into account the fact that different users may use the rating scale differently. The *adjusted* weighted sum, shown in (10c), has been widely used to address this limitation. In this approach, instead of using the absolute values of ratings, the weighted sum uses their deviations from the average rating of the corresponding user. Another way to overcome the differing uses of the rating scale is to deploy *preference-based filtering* [22, 35, 51, 52], which focuses on predicting the *relative* preferences of users instead of absolute rating values, as was pointed out earlier in Section 2.

[3] We use the $r_{c,s} = \varnothing$ notation to indicate that item *s* has not been rated by user *c*.

Various approaches have been used to compute the similarity $sim(c,c')$ between users in collaborative recommender systems. In most of these approaches, the similarity between two users is based on their ratings of items that *both* users have rated. The two most popular approaches are *correlation-* and *cosine-based*. To present them, let S_{xy} be the set of all items co-rated by both users x and y, i.e., $S_{xy} = \{s \in S \mid r_{x,s} \neq \varnothing \ \& \ r_{y,s} \neq \varnothing\}$. In collaborative recommender systems S_{xy} is used mainly as an intermediate result for calculating the "nearest neighbors" of user x and is often computed in a straightforward manner, i.e., by computing the intersection of sets S_x and S_y. However, some methods, such as the graph-theoretic approach to collaborative filtering [4], can determine the nearest neighbors of x without computing S_{xy} for all users y. In the correlation-based approach, the Pearson correlation coefficient is used to measure the similarity [86, 97]:

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)(r_{y,s} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)^2 \sum_{s \in S_{xy}} (r_{y,s} - \bar{r}_y)^2}} \tag{12}$$

In the cosine-based approach [15, 91], the two users x and y are treated as two vectors in m-dimensional space, where $m = |S_{xy}|$. Then, the similarity between two vectors can be measured by computing the cosine of the angle between them:

$$sim(x,y) = \cos(\vec{x},\vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\|_2 \times \|\vec{y}\|_2} = \frac{\sum_{s \in S_{xy}} r_{x,s} r_{y,s}}{\sqrt{\sum_{s \in S_{xy}} r_{x,s}^2} \sqrt{\sum_{s \in S_{xy}} r_{y,s}^2}} \tag{13}$$

where $\vec{x} \cdot \vec{y}$ denotes the dot-product between the vectors $\vec{x}$ and $\vec{y}$. Still another approach to measuring similarity between users uses the *mean squared difference* measure and is described in [97]. Note that different recommender systems may take different approaches in order to implement the user similarity calculations and rating estimations as efficiently as possible. One

common strategy is to calculate *all* user similarities *sim*(x, y) (including the calculation of S_{xy}) in advance and recalculate them only once in a while (since the network of peers usually does not change dramatically in a short time). Then, whenever the user asks for a recommendation, the ratings can be efficiently calculated on demand using pre-computed similarities.

Note, that both the content-based and the collaborative approaches use the same cosine measure from information retrieval literature. However, in content-based recommender systems it is used to measure the similarity between vectors of TF-IDF weights, whereas in collaborative systems it measures the similarity between vectors of the actual user-specified ratings.

Many performance-improving modifications, such as *default voting*, *inverse user frequency*, *case amplification* [15], and *weighted-majority prediction* [27, 72], have been proposed as extensions to these standard correlation-based and cosine-based techniques. For example, the default voting [15] is an extension to the memory-based approaches described above. It was observed that whenever there are relatively few user-specified ratings, these methods would not work well in computing similarity between users x and y since the similarity measure is based on the intersection of the itemsets, i.e., sets of items rated by *both* users x and y. It was empirically shown that the rating prediction accuracy could improve if we assume some default rating value for the missing ratings [15].

Also, while the above techniques traditionally have been used to compute similarities between *users*, [91] proposed to use the same correlation-based and cosine-based techniques to compute similarities between *items* instead and obtain the ratings from them. This idea has been further extended in [29] for top-N item recommendations. In addition, [29, 91] present empirical evidence that item-based algorithms can provide better computational performance than traditional user-based collaborative methods, while at the same time providing comparable or better quality than the best available user-based algorithms.

In contrast to memory-based methods, model-based algorithms [11, 15, 37, 39, 47, 64, 75, 105] use the collection of ratings to learn a *model*, which is then used to make rating predictions. For example, [15] proposes a probabilistic approach to collaborative filtering, where the unknown ratings are calculated as

$$r_{c,s} = E(r_{c,s}) = \sum_{i=0}^{n} i \times \Pr(r_{c,s} = i \mid r_{c,s'}, s' \in S_c) \tag{14}$$

and it is assumed that rating values are integers between 0 and n, and the probability expression is the probability that user c will give a particular rating to item s given that user's ratings of the previously rated items. To estimate this probability, [15] proposes two alternative probabilistic models: cluster models and Bayesian networks. In the first model, like-minded users are clustered into classes. Given the user's class membership, the user ratings are assumed to be independent, i.e., the model structure is that of a naïve Bayesian model. The number of classes and the parameters of the model are learned from the data. The second model represents each item in the domain as a node in a Bayesian network, where the states of each node correspond to the possible rating values for each item. Both the structure of the network and the conditional probabilities are learned from the data. One limitation of this approach is that each user can be clustered into a single cluster, whereas some recommendation applications may benefit from the ability to cluster users into several categories at once. For example, in a book recommendation application, a user may be interested in one topic (e.g., programming) for work purposes and a completely different topic (e.g., fishing) for leisure.

Moreover, [11] proposed a collaborative filtering method in a machine learning framework, where various machine learning techniques (such as artificial neural networks) coupled with feature extraction techniques (such as singular value decomposition – an algebraic technique for reducing dimensionality of matrices) can be used. Both [15] and [11] compare

their respective model-based approaches with standard memory-based approaches and report that in some applications model-based methods outperform memory-based approaches in terms of accuracy of recommendations. However, the comparison in both cases is purely empirical and no underlying theoretical evidence supporting this claim is provided.

There have been several other model-based collaborative recommendation approaches proposed in the literature. A statistical model for collaborative filtering was proposed in [105], and several different algorithms for estimating the model parameters were compared, including K-means clustering and Gibbs sampling. Other collaborative filtering methods include a Bayesian model [20], a probabilistic relational model [37], a linear regression [91], and a maximum entropy model [75]. More recently, a significant amount of research has been done in trying to model the recommendation process using more complex probabilistic models. For instance, [96] view the recommendation process as a sequential decision problem and propose to use Markov decision processes (a well known stochastic technique for modeling sequential decisions) for generating recommendations. Other probabilistic modeling techniques for recommender systems include probabilistic latent semantic analysis [47, 48] and a combination of multinomial mixture and aspect models using generative semantics of Latent Dirichlet Allocation [64]. Similarly, [99] also use probabilistic latent semantic analysis to propose a flexible mixture model that allows modeling the classes of users and items explicitly with two sets of latent variables. Furthermore, [55] use a simple probabilistic model to demonstrate that collaborative filtering is valuable with relatively little data on each user, and that, in certain restricted settings, simple collaborative filtering algorithms are almost as effective as the best possible algorithms in terms of utility.

As in the case of content-based techniques, the main difference between collaborative model-based techniques and heuristic-based approaches is that the model-based techniques

calculate utility (rating) predictions based not on some ad-hoc heuristic rules, but rather based on a *model* learned from the underlying data using statistical and machine learning techniques. A method combining both memory-based and model-based approaches was proposed in [79], where it was empirically demonstrated that the use of this combined approach can provide better recommendations than pure memory-based and model-based collaborative approaches.

A different approach to improving the performance of existing collaborative filtering algorithms was taken in [108], where the input set of user-specified ratings is carefully selected using several techniques that exclude noise, redundancy, and exploit the sparsity of the ratings' data. The empirical results demonstrate the increase in accuracy and efficiency for model-based collaborative filtering algorithms. It is also suggested that the proposed input selection techniques may help the model-based algorithms to address the problem of learning from large databases [108]. Furthermore, among the latest developments, [109] propose a probabilistic approach to collaborative filtering that constitutes yet another way to combine the memory-based and model-based techniques. In particular, [109] propose (a) to use an active learning approach to learn the probabilistic model of each user's preferences and (b) to use the stored user profiles in a mixture model to calculate recommendations. The latter aspect of the proposed approach deploys some of the ideas used in the traditional memory-based algorithms.

The pure collaborative recommender systems do not have some of the shortcomings that content-based systems have. In particular, since collaborative systems use other users' recommendations (ratings), they can deal with any kind of content and recommend any items, even the ones that are dissimilar to those seen in the past. However, collaborative systems have their own limitations [8, 57], as described below.

New user problem. It is the same problem as with content-based systems. In order to make accurate recommendations, the system must first learn the user's preferences from the ratings

that the user makes. Several techniques have been proposed to address this problem. Most of them use *hybrid* recommendation approach, which combines content-based and collaborative techniques. The next section describes hybrid recommender systems in more detail. An alternative approach is presented in [83, 109], where various techniques are explored for determining the best (i.e., most informative to a recommender system) items for a new user to rate. These techniques use strategies that are based on item popularity, item entropy, user personalization, and combinations of the above [83, 109].

New item problem. New items are added regularly to recommender systems. Collaborative systems rely solely on users' preferences to make recommendations. Therefore, until the new item is rated by a substantial number of users, the recommender system would not be able to recommend it. This problem can also be addressed using hybrid recommendation approaches, described in the next section.

Sparsity. In any recommender system, the number of ratings already obtained is usually very small compared to the number of ratings that need to be predicted. Effective prediction of ratings from a small number of examples is important. Also, the success of the collaborative recommender system depends on the availability of a critical mass of users. For example, in the movie recommendation system there may be many movies that have been rated only by few people and these movies would be recommended very rarely, even if those few users gave high ratings to them. Also, for the user whose tastes are unusual compared to the rest of the population there will not be any other users who are particularly similar, leading to poor recommendations [8]. One way to overcome the problem of rating sparsity is to use user profile information when calculating user similarity. That is, two users could be considered similar not only if they rated the same movies similarly, but also if they belong to the same demographic segment. For example, [76] uses gender, age, area code, education, and employment information

of users in the restaurant recommendation application. This extension of traditional collaborative filtering techniques is sometimes called "demographic filtering" [76]. Another approach that also explores similarities among users has been proposed in [49], where the sparsity problem is addressed by applying associative retrieval framework and related spreading activation algorithms to explore transitive associations among consumers through their past transactions and feedback. A different approach for dealing with sparse rating matrices was used in [11, 90], where a dimensionality reduction technique, Singular Value Decomposition (SVD), was used to reduce dimensionality of sparse ratings matrices. SVD is a well-known method for matrix factorization that provides the best lower rank approximations of the original matrix [90].

2.3. Hybrid Methods

Several recommendation systems use a *hybrid* approach by combining collaborative and content-based methods, which helps to avoid certain limitations of content-based and collaborative systems [8, 9, 21, 76, 94, 100, 105]. Different ways to combine collaborative and content-based methods into a hybrid recommender system can be classified as follows: (1) implementing collaborative and content-based methods separately and combining their predictions, (2) incorporating some content-based characteristics into a collaborative approach, (3) incorporating some collaborative characteristics into a content-based approach, and (4) constructing a general unifying model that incorporates both content-based and collaborative characteristics. All of the above approaches have been used by recommender systems researchers, as described below.

1. Combining separate recommenders. One way to build hybrid recommender systems is to implement separate collaborative and content-based systems. Then we can have two different scenarios. First, we can combine the outputs (ratings) obtained from individual recommender systems into one final recommendation using either a linear combination of ratings [21] or a voting scheme [76]. Alternatively, we can use one of the individual recommenders, at any given

moment choosing to use the one that is "better" than others based on some recommendation "quality" metric. For example, the DailyLearner system [13] selects the recommender system that can give the recommendation with the higher level of confidence, while [104] chooses the one whose recommendation is more consistent with past ratings of the user.

2. Adding content-based characteristics to collaborative models. Several hybrid recommender systems, including Fab [8] and the "collaboration via content" approach, described in [76], are based on traditional collaborative techniques but also maintain the content-based profiles for each user. These content-based profiles, and not the commonly rated items, are then used to calculate the similarity between two users. As mentioned in [76], this allows to overcome some sparsity-related problems of a purely collaborative approach, since typically not many pairs of users will have a significant number of commonly rated items. Another benefit of this approach is that users can be recommended an item not only when this item is rated highly by users with similar profiles, but also directly, i.e., when this item scores highly against the user's profile [8]. [40] employs a somewhat similar approach in using the variety of different *filterbots* – specialized content-analysis agents that act as additional participants in a collaborative filtering community. As a result, the users whose ratings agree with some of the filterbots' ratings would be able to receive better recommendations [40]. Similarly, [65] uses a collaborative approach where the traditional user's ratings vector is augmented with additional ratings, which are calculated using a pure content-based predictor.

3. Adding collaborative characteristics to content-based models. The most popular approach in this category is to use some *dimensionality reduction* technique on a *group* of content-based profiles. For example, [100] use latent semantic indexing (LSI) to create a collaborative view of a collection of user profiles, where user profiles are represented by term vectors (as discussed in Section 2.1), resulting in a performance improvement compared to the

pure content-based approach.

4. Developing a single unifying recommendation model. Many researchers have followed this approach in recent years. For instance, [9] propose to use content-based and collaborative characteristics (e.g., the age or gender of users or the genre of movies) in a single rule-based classifier. [80] and [94] propose a unified probabilistic method for combining collaborative and content-based recommendations, which is based on the probabilistic latent semantic analysis [46]. Yet another approach is proposed by [25] and [5], where Bayesian mixed-effects regression models are used that employ Markov chain Monte Carlo methods for parameter estimation and prediction. In particular, [5] uses the profile information of users *and* items in a single statistical model that estimates unknown ratings r_{ij} for user *i* and item *j*:

$$r_{ij} = x_{ij}\mu + z_i\gamma_j + w_j\lambda_i + e_{ij}, \quad e_{ij} \sim N(0,\sigma^2), \quad \lambda_i \sim N(0,\Lambda), \quad \gamma_j \sim N(0,\Gamma). \qquad (15)$$

where $i = 1,\ldots,I$ and $j = 1,\ldots,J$ represent users and items respectively, and e_{ij}, λ_i, and γ_j are random variables taking into effect noise, unobserved sources of user heterogeneity and item heterogeneity respectively. Also, x_{ij} is a matrix containing user and item characteristics, z_i is a vector of user characteristics, and w_j is a vector of item characteristics. The unknown parameters of this model are μ, σ^2, Λ, and Γ, and they are estimated from the data of already known ratings using Markov chain Monte Carlo methods. In summary, [5] uses user attributes $\{z_i\}$ constituting a part of a user profile, item attributes $\{w_j\}$ constituting a part of an item profile and their interactions $\{x_{ij}\}$ to estimate the rating of an item.

Hybrid recommendation systems can also be augmented by knowledge-based techniques [17], such as case-based reasoning, in order to improve recommendation accuracy and to address some of the limitations (e.g., new user, new item problems) of traditional recommender systems. For example, knowledge-based recommender system Entrée [17] uses some domain knowledge

about restaurants, cuisines, and foods (e.g., that "seafood" is not "vegetarian") to recommend restaurants to its users. The main drawback of knowledge-based systems is a need for knowledge acquisition – a well-known bottleneck for many artificial intelligence applications. However, knowledge-based recommendation systems have been developed for application domains where domain knowledge is readily available in some structured machine-readable form, e.g., as an ontology. For example, Quickstep and Foxtrot systems [66] use research paper topic ontology to recommend online research articles to the users.

Moreover, several papers, such as [8, 65, 76, 100], empirically compare the performance of the hybrid with the pure collaborative and content-based methods and demonstrate that the hybrid methods can provide more accurate recommendations than pure approaches.

2.4. Summary and Conclusions

As described in Sections 2.1-2.3, there has been much research done on recommendation technologies over the past several years that have used a broad range of statistical, machine learning, information retrieval and other techniques and that significantly advanced the state-of-art in comparison to early recommender systems that utilized collaborative- and content-based heuristics. As was discussed above, recommender systems can be categorized as being (a) *content-based, collaborative*, or *hybrid*, based on the recommendation approach used, and (b) *heuristic-based* or *model-based* based on the types of recommendation techniques used for the rating estimation. We use these two orthogonal dimensions to classify the recommender systems research in the 2×3 matrix presented in Table 2.

The recommendation methods described in this section have performed well in several applications, including the ones for recommending books, CDs, and news articles [64, 88], and some of these methods are used in the "industrial-strength" recommender systems, such as the ones deployed at Amazon [61], MovieLens [67], and VERSIFI Technologies (formerly

AdapiveInfo.com) [14]. However, both collaborative and content-based methods have certain limitations described earlier in this section. Moreover, in order to provide better recommendations and to be able to use recommender systems in arguably more complex types of applications, such as recommending vacations or certain types of financial services, most of the methods reviewed in this section would need significant extensions. For example, even for a traditional movie recommendation application, [3] showed that, by extending the traditional memory-based collaborative filtering approach to take into the consideration the *contextual* information, such as when, where and with whom a movie is seen, the resulting recommender system could outperform the pure traditional collaborative filtering method. Many real-life recommendation applications, including several business applications, such as the ones described above, are arguably more complex than a movie recommender system, and would require taking more factors into the recommendation consideration. Therefore, the need to develop more advanced recommendation methods is even more pressing for such types of applications. In the next section, we review various ways to extend recommendation methods in order to support more complex types of recommendation applications.

3. Extending Capabilities of Recommender Systems

Recommender systems, as described in Section 2 and summarized in Table 2, can be extended in several ways that include improving the understanding of users and items, incorporating the contextual information into the recommendation process, supporting multi-criteria ratings, and providing more flexible and less intrusive types of recommendations. Such more comprehensive models of recommender systems can provide better recommendation capabilities. In the remainder of this section we describe the proposed extensions and also identify various research opportunities for developing them.

Recommendation Approach	Recommendation Technique	
	Heuristic-based	Model-based
Content-based	Commonly used techniques: • TF-IDF (information retrieval) • Clustering Representative research examples: • Lang 1995 • Balabanovic & Shoham 1997 • Pazzani & Billsus 1997	Commonly used techniques: • Bayesian classifiers • Clustering • Decision trees • Artificial neural networks Representative research examples: • Pazzani & Billsus 1997 • Mooney et al. 1998 • Mooney & Roy 1999 • Billsus & Pazzani 1999, 2000 • Zhang et al. 2002
Collaborative	Commonly used techniques: • Nearest neighbor (cosine, correlation) • Clustering • Graph theory Representative research examples: • Resnick et al. 1994 • Hill et al. 1995 • Shardanand & Maes 1995 • Breese et al. 1998 • Nakamura & Abe 1998 • Aggarwal et al. 1999 • Delgado & Ishii 1999 • Pennock & Horwitz 1999 • Sarwar et al. 2001	Commonly used techniques: • Bayesian networks • Clustering • Artificial neural networks • Linear regression • Probablistic models Representative research examples: • Billsus & Pazzani 1998 • Breese et al. 1998 • Ungar & Foster 1998 • Chien & George 1999 • Getoor & Sahami 1999 • Pennock & Horwitz 1999 • Goldberg et al. 2001 • Kumar et al. 2001 • Pavlov & Pennock 2002 • Shani et al. 2002 • Yu et al. 2002, 2004 • Hofmann 2003, 2004 • Marlin 2003 • Si & Jin 2003
Hybrid	Combining content-based and collaborative components using: • Linear combination of predicted ratings • Various voting schemes • Incorporating one component as a part of the heuristic for the other Representative research examples: • Balabanovic & Shoham 1997 • Claypool et al. 1999 • Good et al. 1999 • Pazzani 1999 • Billsus & Pazzani 2000 • Tran & Cohen 2000 • Melville et al. 2002	Combining content-based and collaborative components by: • Incorporating one component as a part of the model for the other • Building one unifying model Representative research examples: • Basu et al. 1998 • Condliff et al. 1999 • Soboroff & Nicholas 1999 • Ansari et al. 2000 • Popescul et al. 2001 • Schein et al. 2002

Table 2: Classification of recommender systems research.

3.1. Comprehensive understanding of users and items

As was pointed out in [2, 8, 54, 105], most of the recommendation methods produce ratings that are based on a limited understanding of users and items as captured by user and item profiles and do not take full advantage of the information in the user's transactional histories and other

available data. For example, classical collaborative filtering methods [45, 86, 97] do not use user and item profiles at all for the recommendation purposes and rely exclusively on the ratings information to make recommendations. Although there has been some progress made on incorporating user and item profiles into some of the methods since the earlier days of recommender systems [13, 76, 79], still these profiles tend to be quite simple and do not utilize some of the more advanced profiling techniques. In addition to using traditional profile features, such as keywords and simple user demographics [69, 77], more advanced profiling techniques based on data mining rules [1, 34], sequences [63], and signatures [26] that describe user's interests can be used to build user profiles. Also, in addition to using the traditional item profile features, such as keywords [9, 76], similar advanced profiling techniques can also be used to build comprehensive item profiles. With respect to recommender systems, advanced profiling techniques that are based on data mining have been used mainly in the context of Web usage analysis [59, 68, 110], i.e., to discover *navigational* Web usage patterns (i.e., page view sequences) of users in order to provide better Web site recommendations; however, such techniques have not been widely adopted in rating-based recommender systems.

Once user and item profiles are built, the most general ratings estimation function can be defined in terms of these profiles and the previously specified ratings as follows. Let profile of user i be defined as a vector of p features, i.e., $\vec{c}_i = (a_{i1}, \ldots, a_{ip})$. Also, let profile of item j be defined as a vector of r features, i.e., $\vec{s}_j = (b_{j1}, \ldots, b_{jr})$. We deliberately did not define precisely the meanings of features a_{ij} and b_{kl} because they can mean different concepts in different applications, such as numbers, categories, rules, sequences, etc. Also, let $\vec{c}$ be a vector of all user profiles, i.e., $\vec{c} = (\vec{c}_1, \ldots, \vec{c}_m)$, and let $\vec{s}$ be a vector of all item profiles, i.e., $\vec{s} = (\vec{s}_1, \ldots, \vec{s}_n)$. Then the most general rating estimation procedure can be defined as

$$r'_{ij} = \begin{cases} r_{ij}, & \text{if } r_{ij} \neq \varnothing \\ u_{ij}(R, \vec{c}, \vec{s}), & \text{if } r_{ij} = \varnothing \end{cases} \quad (16)$$

that estimates each unknown rating $r'_{ij} = u_{ij}(R, \vec{c}, \vec{s})$ in terms of known ratings $R = \{ r_{ij} \neq \varnothing \}$, user profiles $\vec{c}$, and item profiles $\vec{s}$. We can use various methods for estimating utility function u_{ij}, including various heuristics, nearest neighbor classifiers, decision trees, spline methods, radial basis functions, regressions, and neural networks. Moreover, we would like to point out that equation (16) presents the most general model that depends on a whole range of inputs, including the characteristics of user *i* ($\vec{c}_i$) and possibly other users $\vec{c} = (\vec{c}_1, \ldots, \vec{c}_m)$, characteristics of item *j* ($\vec{s}_j$), and possibly other items $\vec{s} = (\vec{s}_1, \ldots, \vec{s}_n)$, ratings (preferences) R_i expressed by user *i* and ratings (preferences) expressed by all other users $R = \{r_{ij} \neq \varnothing\}$. Therefore, function u_{ij} clearly subsumes collaborative, content-based and hybrid methods discussed in Section 2. However, most of the existing recommender systems make function u_{ij} dependent only on a (small) subset of the whole input space *R*, $\vec{c}$, and $\vec{s}$. For example, function u_{ij} for traditional memory-based collaborative filtering methods does not depend on inputs $\vec{c}$ and $\vec{s}$ and restricts *R* only to column R_j and usually only to the set of *N* nearest neighbors r_{ij} for column R_j.[4]

An interesting research problem would be to extend the attribute-based profiles, as defined by $\vec{c}$ and $\vec{s}$, to utilize more advanced profiling techniques described above, such as rule-, sequence-, and signature-based methods.

3.2. Extensions for Model-Based Recommendation Techniques

As discussed in Section 2, some of the model-based approaches provide rigorous rating estimation methods utilizing various statistical and machine learning techniques. However, other areas of mathematics and computer science, such as *mathematical approximation theory* [16, 73,

[4] Actually, the situation is a little more complicated than this because estimation of nearest neighbors may involve other values of matrix *R* for some of the collaborative filtering methods.

81], can also contribute to developing better rating estimation methods defined by equation (16). One example of an approximation-based approach to defining function u_{ij} in (16) constitutes *radial basis functions* [16, 30, 92] that are defined as follows. Given a set of points $X = \{x_1, \ldots, x_m\}$ (where $x_i \in \mathbb{R}^N$) and the values of an unknown function f (e.g., the rating function) at these points, i.e., $f(x_1)$, …, $f(x_m)$, a radial basis function $r_{f,X}$ estimates the values of f in the whole $\mathbb{R}^N$, given $r_{f,X}(x_i) = f(x_i)$ for all $i = 1, \ldots, m$, as

$$r_{f,X}(x) = \sum_{i=1}^{m} \alpha_i \phi(\|x - x_i\|) \tag{17}$$

where $\{\alpha_1, \ldots, \alpha_m\}$ are coefficients from $\mathbb{R}$, $\|x\|$ is a norm (e.g., L_2) and ϕ is a positive definite function, i.e., a function satisfying the condition

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j \phi(\|x_i - x_j\|) > 0 \tag{18}$$

for all distinct points x_1, …, x_m in $\mathbb{R}^N$ and all the coefficients α_1, …, α_m from $\mathbb{R}$. Then a well-known theorem [92] states that if ϕ is a positive definite function then there exists a unique function $r_{f,X}$ of the form (17) satisfying the conditions $r_{f,X}(x_i) = f(x_i)$ for all $i = 1, \ldots, m$. Some popular examples of positive definite functions ϕ are:

1. $\phi(r) = r^{\beta}$, where $\beta > 0$ is a positive odd number;
2. $\phi(r) = r^k \log(r)$, where $k \in \mathbb{N}$ (thin-plate splines);
3. $\phi(r) = e^{-\alpha r^2}$ where $\alpha > 0$ (Gaussian).

One of the advantages of radial basis functions is that they have been extensively studied in the approximation theory, and their theoretical properties and utilization of radial basis functions in many practical applications have been understood very well [16, 92]. Therefore, it should be interesting to apply them for estimating unknown ratings in recommender systems.

One caveat with using radial basis functions in recommender systems, though, is that the

recommendation space $\vec{c} \times \vec{s}$ does not usually constitute an *N*-dimensional Euclidean space $\mathbb{R}^N$. Therefore, one research challenge is to extend radial basis methods from the real numbers to other domains and apply them to recommender systems problems. The applicability of other approximation methods for estimating u_{ij} in (16) constitutes another interesting research topic.

3.3. Multidimensionality of recommendations

Current generation of recommender systems operates in the two-dimensional *User*×*Item* space. That is, they make their recommendations based only on the user and item information and do not take into the consideration additional *contextual* information that may be crucial in some applications. However, in many situations the utility of a certain product to a user may depend significantly on time (e.g., the time of the year, such as season or month, or the day of the week). It may also depend on the person(s) with whom the product will be consumed or shared and under which circumstances. In such situations it may not be sufficient to simply recommend items to users; the recommender system must take additional contextual information, such as time, place, and the company of a user, into the consideration when recommending a product. For example, when recommending a vacation package, the system should also consider the time of the year, with whom the user plans to travel, traveling conditions and restrictions at that time, and other contextual information. As another example, a user can have significantly different preferences for the types of movies she wants to see when she is going out to a movie theater with a boyfriend on a Saturday night as opposed to watching a rental movie at home with her parents on a Wednesday evening. As was argued in [2], it is important to extend traditional two-dimensional *User*×*Item* recommendation methods to multi-dimensional settings. In addition, [43] argued that the inclusion of the knowledge about user's task into the recommendation algorithm in certain applications can lead to better recommendations.

In order to take into the consideration the contextual information, [2] propose to define the utility (or ratings) function over a multidimensional space $D_1 \times \ldots \times D_n$ (as opposed to the traditional 2-dimensional *User*×*Item* space) as

$$u : D_1 \times \ldots \times D_n \rightarrow R \tag{19}$$

Then a recommendation problem is defined by selecting certain "what" dimensions $D_{i1}, \ldots, D_{ik}$ ($k < n$) and certain "for whom" dimensions $D_{j1}, \ldots, D_{jl}$ ($l < n$) that do not overlap, i.e., $\{D_{i1}, \ldots, D_{ik}\} \bigcap \{D_{j1}, \ldots, D_{jl}\} = \varnothing$, and recommending for each tuple $(d_{j1}, \ldots, d_{jl}) \in D_{j1} \times \ldots \times D_{jl}$ the tuple $(d_{i1}, \ldots, d_{ik}) \in D_{i1} \times \ldots \times D_{ik}$ that maximizes the utility $u(d_1, \ldots, d_n)$, i.e.,

$$\forall (d_{j1}, \ldots, d_{jl}) \in D_{j1} \times \ldots \times D_{jl}, \qquad (d_{i1}, \ldots, d_{ik}) = \underset{\substack{(d'_{i1}, \ldots, d'_{ik}) \in D_{i1} \times \ldots \times D_{ik} \\ (d'_{j1}, \ldots, d'_{jl}) = (d_{j1}, \ldots, d_{jl})}}{\arg\max} u(d'_1, \ldots, d'_n) \tag{20}$$

For example, in the case of a movie recommender system one needs to consider not only characteristics of the movie d_1 and of the person who wants to see the movie d_2, but also such contextual information as (a) d_3: where and how the movie will be seen (e.g., in the movie theater, at home on TV, on video or DVD), (b) d_4: with whom the movie will be seen (e.g., alone, with girlfriend/boyfriend, friends, parents, etc.), and (c) d_5: when will the movie be seen (e.g., on weekdays or weekends, in the morning/afternoon/evening, during the opening night, etc.). As discussed earlier, each of the components d_1, d_2, d_3, d_4, d_5 can be defined as a *vector* of its characteristics, and the overall utility function $u(d_1, d_2, d_3, d_4, d_5)$ can be quite complex and take into consideration various interaction effects among vectors d_1, d_2, d_3, d_4, d_5.

As was argued in [2, 3], many of the two-dimensional recommendation algorithms cannot be directly extended to the multidimensional case. Furthermore, [3] proposes a *reduction-based* recommendation approach which uses only the ratings that pertain to the context of the user-specified criteria in which a recommendation is made. For example, to recommend a movie to a

person who wants to see it in a movie theater on a Saturday night, the reduction-based approach would use only the available ratings of the movies seen in the movie theaters over the weekends, if it is determined from the data that the *place* and the *time of the week* dimensions affect the moviegoers' behavior. By selecting only the ratings relevant to a recommendation context, the reduction-based approach projects the multi-dimensional cube of ratings on the two primary *User* and *Item* dimensions. Then any standard two-dimensional recommendation method described in Section 2 can be used to produce a recommendation. Since these recommendations are based only on the context-specific set of ratings, this amounts to building a *local* model producing context-specific recommendations.

Another possible approach to producing multi-dimensional recommendations would be to deploy the hierarchical Bayesian method presented in [5], which can be extended from 2- to multi-dimensional case as follows. Instead of considering the two-dimensional case, as defined in (15), where user characteristics d_1 are defined with vector z_i and item characteristics d_2 with vector w_j, we can also add contextual dimensions $d_3, \ldots, d_n$, where $d_i = (d_{i1}, \ldots, d_{ix_i})$ is a vector of characteristics for dimension D_i. Then the rating function $r = u(d_1, d_2, \ldots, d_n)$ is extended from (15) to the linear combination of $d_1, d_2, \ldots, d_n$ and also includes interaction effects among these dimensions (i.e., interaction effects, as defined by matrix $\{ x_{ij} \}$ in (15), should be extended to include other dimensions). One of the research challenges is to make these extensions scalable for large values of n.

3.4. Multi-criteria ratings

Most of the current recommender systems deal with single-criterion ratings, such as ratings of movies and books. However, in some applications, such as restaurant recommenders, it is crucial to incorporate multi-criteria ratings into recommendation methods. For example, many

restaurant guides, such as Zagat's Guide, provide three criteria for restaurant ratings: food, decor and service. Although multi-criteria ratings have not yet been examined in the recommender systems literature, they have been extensively studied in the Operations Research community [33, 102]. Typical solutions to the multi-criteria optimization problems include (a) finding Pareto optimal solutions, (b) taking a linear combination of multiple criteria and reducing the problem to a single-criterion optimization problem, (c) optimizing the most important criterion and converting other criteria to constraints, (d) consecutively optimizing one criterion at a time, converting an optimal solution to constraint(s) and repeating the process for other criteria. An example of the latter approach is the method of successive concessions [102].

To illustrate how some of these methods can be used in recommender systems, consider the application of approach (c) to the problem of recommending restaurants *r* to user *c* based on the user's criteria of food quality $f_c(r)$, décor $d_c(r)$, and service $s_c(r)$. We can take food quality $f_c(r)$ to be the primary criterion and use others as constraints, i.e., we want to find restaurants *r* that maximize $f_c(r)$, subject to the constraints $d_c(r) > \alpha_c$ and $s_c(r) > \beta_c$, where α_c and β_c are minimal ratings for décor and service (e.g., user *c* will not go to any restaurant having décor and service ratings below 10, out of possible 30, regardless of the quality of food there). This problem is complicated by the fact that we usually will not have the user's decor $d_c(r)$ and service $s_c(r)$ ratings for all the restaurants. Then the task of a recommender system is to estimate unknown ratings $d'_c(r)$ and $s'_c(r)$, e.g., using the rating estimation methods described in Section 2, and find all the restaurants *r* satisfying constraints $d'_c(r) > \alpha_c$ and $s'_c(r) > \beta_c$. Once we find all the restaurants satisfying the constraints with these estimated ratings, we can use *those* restaurants in search for the maximum of $f_c(r)$. However, as with décor and service

ratings, we might not have the user's food ratings $f_c(r)$ for all such restaurants and, thus, will also need to use rating estimation procedure for $f_c(r)$ before making any recommendations.

We believe that the problem of finding Pareto-optimal solution set and the iterative method of consecutive single criterion optimizations for multi-criteria recommendation problems mentioned above should also constitute interesting and challenging problems.

3.5. Non-intrusiveness

Many recommender systems are intrusive in the sense that they require explicit feedback from the user and often at a significant level of user involvement. For example, before recommending any newsgroup articles, the system needs to acquire ratings of previously read articles, and often many of them. Since it is impractical to elicit many ratings of these articles from the user, some recommender systems use non-intrusive rating determination methods where certain proxies are used to estimate real ratings. For example, the amount of time a user spends reading a newsgroup article can serve as a proxy of the article's rating given by this user. Some non-intrusive methods of getting user feedback are presented in [18, 53, 66, 74, 94]. However, non-intrusive ratings (such as time spent reading an article) are often inaccurate and cannot fully replace explicit ratings provided by the user. Therefore, the problem of minimizing intrusiveness while maintaining certain levels of accuracy of recommendations needs to be addressed by the recommender systems researchers.

One way to explore the intrusiveness problem is to determine an optimal number of ratings the system should ask from a new user. For example, before recommending any movies, MovieLens.org first asks the user to rate a predefined number of movies (e.g., 20). This request incurs certain costs on the end-user that can be modeled in various ways, the simplest model being a fixed-cost model (i.e., the cost of rating each movie is *C* and the cost of rating *n* movies

is $C \cdot n$). Then the intrusiveness problem can be formulated as an optimization problem that tries to find an optimal number of initial rating requests n as follows. Each additional rating supplied by the user increases the accuracy of recommendations (or any other effectiveness measure) and, therefore, results in certain benefits for the user. One interesting intrusiveness-related research problem would be to develop formal models for defining and measuring benefit $B(n)$ of supplying n initial ratings in terms of the increased accuracy of predictions based on these ratings. Once it is known how to measure benefits $B(n)$ (e.g., by measuring the predictive accuracy of a recommender system), we need to determine an optimal number of initial ratings n that maximizes expression $B(n) - C \cdot n$. Clearly, optimal value of n is reached when marginal benefits are equal to marginal costs, i.e., when $\Delta B(n) = C$. The optimal solution should exist under the assumption that $B(n)$ is a monotonically increasing function in n with decreasing marginal benefits $\Delta B(n)$ that asymptotically converge to zero.

Another interesting research opportunity lies in developing marginal cost models that are more advanced than the fixed-cost model described above and that can potentially include cost/benefit analysis of using *both* implicit and explicit ratings in a recommender system.

Finally, the issue of incrementally selecting good training data for modeling purposes is the problem of *active learning*, which is a fairly well-studied area in the machine learning literature, and numerous approaches have been proposed to addressing this problem [23, 24, 36, 58]. We believe that applying active learning methods to address the non-intrusiveness issue constitutes another interesting research opportunity.

3.6. Flexibility

Most of the recommendation methods are inflexible in the sense that they are "hard-wired" into the systems by the vendors and therefore support only a predefined and fixed set of recommendations. Therefore, the end-user cannot customize recommendations according to his

or her needs in real time. This problem has been identified in [2], and Recommendation Query Language (RQL) has been proposed to address it [2]. RQL is an SQL-like language for expressing flexible user-specified recommendation requests. For example, the request "recommend to each user from New York the best three movies that are longer than two hours" can be expressed in RQL as:

RECOMMEND *Movie* **TO** *User*
BASED ON *Rating*
SHOW TOP 3
FROM *MovieRecommender*
WHERE *Movie.Length* > 120 **AND** *User.City* = "New York".

Also, most of the recommender systems recommend only individual items to individual users and do not deal with aggregation. However, it is important to be able to provide *aggregated* recommendations in a number of applications, such as recommend *brands* or *categories* of products to certain *segments* of users. For example, a travel-related recommender system may want to recommend vacations in Florida (category of products) to the undergraduate students from the Northeast (user segment) during the spring break. One way to support aggregated recommendations is by utilizing the OLAP-based approach [19] to multidimensional recommendations. OLAP-based systems naturally support aggregation hierarchies, and the initial approaches to deploying OLAP-based methods in recommender systems are presented in [2, 3]. However, more work is required to develop a more comprehensive understanding of how to use the OLAP approach in recommender systems, and this constitutes an interesting and challenging research problem.

3.7. Effectiveness of recommendations

The problem of developing good metrics to measure effectiveness of recommendations has been extensively addressed in the recommender systems literature. Some examples of this work include [41, 44, 69, 107]. In most of the recommender systems literature, the performance

evaluation of recommendation algorithms is usually done in terms of the *coverage* and *accuracy* metrics. Coverage measures the percentage of items for which a recommender system is capable of making predictions [41]. Accuracy measures can be either *statistical* or *decision-support* [41]. Statistical accuracy metrics mainly compare the estimated ratings (e.g., as defined in (16)) against the actual ratings R in the $User \times Item$ matrix, and include Mean Absolute Error (MAE), root mean squared error, and correlation between predictions and ratings. Decision-support measures determine how well a recommender system can make predictions of high-relevance items (i.e., items that would be rated highly by the user). They include classical IR measures of precision (the percentage of truly "high" ratings among those that were predicted to be "high" by the recommender system), recall (the percentage of correctly predicted "high" ratings among all the ratings known to be "high"), F-measure (a harmonic mean of precision and recall), and Receiver Operating Characteristic (ROC) measure demonstrating the tradeoff between *true positive* and *false positive* rates in recommender systems [41].

Although popular, these empirical evaluation measures have certain limitations. One limitation is that these measures are typically performed on test data that the users *chose* to rate. However, items that users choose to rate are likely to constitute a skewed sample, e.g., users may rate mostly the items that they like. In other words, the empirical evaluation results typically only show how accurate the system is on items the user decided to rate, whereas the ability of the system to properly evaluate a random item (which it should be able to do during its normal real-life use) is not tested. Understandably, it is expensive and time-consuming to conduct controlled experiments with users in the recommender systems settings, therefore, the experiments that test recommendation quality on an unbiased random sample are rare, e.g., [69]. However, the high-quality experiments are necessary in order to truly understand the benefits and limitations of the proposed recommendation techniques.

In addition, although crucial for measuring accuracy of recommendations, the technical measures mentioned earlier often do not capture adequately "usefulness" and "quality" of recommendations. For example, as [107] observe for a supermarket application, recommending obvious items (such as milk or bread) that the consumer will buy anyway will produce high accuracy rates; however, it will not be very helpful to the consumer. Therefore, it is also important to develop economics-oriented measures that capture the business value of recommendations, such as return on investments (ROI) and customer lifetime value (LTV) measures [32, 88, 95]. Developing and studying the measures that would remedy the limitations described in this section constitutes an interesting and important research topic.

3.8. Other Extensions

Other important research issues that have been explored in recommender systems literature include explainability [12, 42], trustworthiness [28], scalability [4, 39, 91, 93], and privacy [82, 93] issues of recommender systems. However, we will not review this work and will not discuss research opportunities in these areas because of the space limitation.

4. Conclusions

Recommender systems made a significant progress over the last decade when numerous content-based, collaborative and hybrid methods were proposed and several "industrial-strength" systems have been developed. However, despite all these advances, the current generation of recommender systems surveyed in this paper still requires further improvements to make recommendation methods more effective in a broader range of applications. In this paper, we reviewed various limitations of the current recommendation methods and discussed possible extensions that can provide better recommendation capabilities. These extensions include, among others, the improved modeling of users and items, incorporation of the contextual information into the recommendation process, support for multi-criteria ratings, and provision of

a more flexible and less intrusive recommendation process. We hope that the issues presented in this paper would advance the discussion in the recommender systems community about the next generation of recommendation technologies.

References

1. Adomavicius, G. and A. Tuzhilin. Expert-driven validation of rule-based user models in personalization applications. *Data Mining and Knowledge Discovery*, 5(1/2):33-58, 2001a.
2. Adomavicius, G. and A. Tuzhilin. Multidimensional recommender systems: a data warehousing approach. In *Proc. of the 2nd Intl. Workshop on Electronic Commerce (WELCOM'01). Lecture Notes in Computer Science, vol. 2232*, Springer, 2001b.
3. Adomavicius, G., R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. *ACM Transactions on Information Systems*, 23(1), January 2005.
4. Aggarwal, C. C., J. L. Wolf, K-L. Wu, and P. S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 1999.
5. Ansari, A., S. Essegaier, and R. Kohli. Internet recommendations systems. *Journal of Marketing Research*, pages 363-375, August 2000.
6. Armstrong, J. S. *Principles of Forecasting – A Handbook for Researchers and Practitioners,* Kluwer Academic Publishers, 2001.
7. Baeza-Yates, R., B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
8. Balabanovic, M. and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66-72, 1997.
9. Basu, C., H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98-08*. AAAI Press, 1998.
10. Belkin, N. and B. Croft. Information filtering and information retrieval. *Communications of the ACM*, 35(12):29-37, 1992.
11. Billsus, D. and M. Pazzani. Learning collaborative information filters. In *International Conference on Machine Learning*, Morgan Kaufmann Publishers, 1998.
12. Billsus, D. and M. Pazzani. A Personal News Agent That Talks, Learns and Explains. In *Proceedings of the Third Annual Conference on Autonomous Agents*, 1999.
13. Billsus, D. and M. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2-3):147-180, 2000.
14. Billsus, D., C. A. Brunk, C. Evans, B. Gladish, and M. Pazzani. Adaptive interfaces for ubiquitous web access. *Commnications of the ACM*, 45(5):34-38, 2002.
15. Breese, J. S., D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, July 1998.
16. Buhmann, M. D. Approximation and interpolation with radial functions. In *Multivariate Approximation and Applications*. Eds. N. Dyn, D. Leviatan, D. Levin, and A. Pinkus. Cambridge University Press, 2001.
17. Burke, R. Knowledge-based recommender systems. In A. Kent (ed.), *Encyclopedia of Library and Information Systems*. Volume 69, Supplement 32. Marcel Dekker, 2000.

18. Caglayan, A., M. Snorrason, J. Jacoby, J. Mazzu, R. Jones, and K. Kumar. Learn Sesame – a learning agent engine. *Applied Artificial Intelligence*, 11:393-412, 1997.
19. Chaudury, S. and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65-74, 1997.
20. Chien, Y-H. and E. I. George. A bayesian model for collaborative filtering. In *Proc. of the 7th International Workshop on Artificial Intelligence and Statistics*, 1999.
21. Claypool, M., A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *ACM SIGIR'99. Workshop on Recommender Systems: Algorithms and Evaluation*, August 1999.
22. Cohen, W. W., R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Articial Intelligence Research*, 10:243-270, 1999.
23. Cohn, D., L. Atlas, and R. Ladner. Improving Generalization with Active Learning. *Machine Learning*, 15(2):201-221, 1994.
24. Cohn, D., Z. Ghahramani, and M. Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, 4:129-145, 1996.
25. Condliff, M., D. Lewis, D. Madigan, and C. Posse. Bayesian mixed-effects models for recommender systems. In *ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation*, August 1999.
26. Cortes, C., K. Fisher, D. Pregibon, A. Rogers, and F. Smith. Hancock: a language for extracting signatures from data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.
27. Delgado, J. and N. Ishii. Memory-based weighted-majority prediction for recommender systems. In *ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.
28. Dellarocas, C. The Digitization of Word of Mouth: Promise and Challenges of Online Feedback Mechanisms. *Management Science*, 49(10):1407-1424, 2003.
29. Deshpande, M. and G. Karypis. Item-Based Top-*N* Recommendation Algorithms. *ACM Transactions on Information Systems*, 22(1):143-177, 2004.
30. Duchon, J. Splines minimizing rotation-invariate semi-norms in Sobolev spaces. In *Constructive Theory of Functions of Several Variables*, ed. W. Schempp & Zeller, pp. 85-100, Springer, 1979.
31. Duda, R. O., P. E. Hart, and D. G. Stork. *Pattern Classification*, John Wiley & Sons, 2001.
32. Dwyer, F. R. Customer Lifetime Valuation to Support Marketing Decision Making. *Journal of Direct Marketing*, Vol 3(4), 1989.
33. Ehrgott, M. *Multicriteria Optimization*. Springer Verlag, September 2000.
34. Fawcett, T., and F. Provost. Combining data mining and machine learning for efficient user profiling. In *Proceedings of the Second International Conference On Knowledge Discovery and Data Mining (KDD-96)*, 1996.
35. Freund, Y., R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proc. of the 15th Intl. Conference on Machine Learning*, 1998.
36. Freund, Y., H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133-168, 1997.
37. Getoor, L. and M. Sahami. Using probabilistic relational models for collaborative filtering. In *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*, August 1999.
38. Goldberg, D., D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61-70, 1992.
39. Goldberg, K., T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time

collaborative filtering algorithm. *Information Retrieval Journal*, 4(2):133-151, July 2001.
40. Good, N., J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. L. Herlocker, and J. Riedl. Combining Collaborative Filtering with Personal Agents for Better Recommendations. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI-99)*, pp. 439-446, Orlando, Florida, July 1999,
41. Herlocker, J. L., J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99).* 1999.
42. Herlocker, J. L., J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 2000.
43. Herlocker, J. L. and J. A. Konstan. Content-Independent Task-Focused Recommendation. *IEEE Internet Computing*, 5(6):40-47, 2001.
44. Herlocker, J. L., J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 22(1):5-53, 2004.
45. Hill, W., L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of CHI'95*.
46. Hofmann, T. Probabilistic Latent Semantic Analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 289-296, 1999.
47. Hofmann, T. Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis. In *Proc. of the 26th Annual International ACM SIGIR Conference*, Toronto, Canada, 2003.
48. Hofmann, T. Latent Semantic Models for Collaborative Filtering. *ACM Transactions on Information Systems*, 22(1):89-115, 2004.
49. Huang, Z., H. Chen, and D. Zeng. Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering. *ACM Transactions on Information Systems*, 22(1):116-142, 2004.
50. Hull, D. A. The TREC-7 Filtering Track: Description and Analysis. In *Proceedings of the 7th Text Retrieval Conference (TREC-7)*, pp., 1999.
51. Jin, R., L. Si, and C. Zhai. Preference-based Graphic Models for Collaborative Filtering. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI 2003)*, Acapulco, Mexico, August 2003a.
52. Jin, R., L. Si, C. Zhai, and J. Callan. Collaborative Filtering with Decoupled Models for Preferences and Ratings. In *Proc. of the 12th International Conference on Information and Knowledge Management (CIKM 2003)*, New Orleans, LA, November 2003b.
53. Konstan, J. A., B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77-87, 1997.
54. Konstan, J. A., J. Riedl, A. Borchers, and J. L. Herlocker. Recommender systems: a GroupLens perspective. In *Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98-08*. AAAI Press, 1998.
55. Kumar, R., P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation Systems: A Probabilistic Analysis. *Journal of Computer and System Sciences*, 63(1):42-61, 2001.
56. Lang, K. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, 1995.
57. Lee, W. S. Collaborative learning for recommender systems. In *Proccedings of the International Conference on Machine Learning*, 2001.

58. Lewis, D. and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of 11th International Conference on Machine Learning*, pp. 148-156, 1994.
59. Li, J. and O. R. Zaïane. Combining Usage, Content and Structure Data to Improve Web Site Recommendation. In *Proceedings of the 5th International Conference on Electronic Commerce and Web Technologies (EC-Web 04)*, pp. 305-315, Zaragoza, Spain, 2004.
60. Lilien, G. L., P. Kotler, K. S. Moorthy. *Marketing Models*, Prentice Hall, 1992.
61. Linden, G., B. Smith, and J. York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing,* Jan.-Feb. 2003.
62. Littlestone, N. and M. Warmuth. The Weighted Majority Algorithm. *Information and Computation*, 108(2):212-261, 1994.
63. Mannila, H., H. Toivonen, and A. I. Verkamo. Discovering Frequent Episodes in Sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, 1995.
64. Marlin, B. Modeling User Rating Profiles for Collaborative Filtering. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems (NIPS'03)*, 2003.
65. Melville, P., R. J. Mooney, and R. Nagarajan. Content-Boosted Collaborative Filtering for Improved Recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, Edmonton, Canada, 2002.
66. Middleton, S. E., N. R. Shadbolt, and D. C. de Roure. Ontological User Profiling in Recommender Systems. *ACM Transactions on Information Systems*, 22(1):54-88, 2004.
67. Miller, B. N., I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System. In *Proceedings of the International Conference on Intelligent User Interfaces*, Miami, Florida, 2003.
68. Mobasher, B., H. Dai, T. Luo, and M. Nakagawa. Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. *Data Mining and Knowledge Discovery*, 6(1):61-82, 2002.
69. Mooney, R. J. and L. Roy Content-based book recommending using learning for text categorization. In *ACM SIGIR'99. Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.
70. Mooney, R. J., P. N. Bennett, and L. Roy. Book recommending using text categorization with extracted information. In *Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98-08*. AAAI Press, 1998.
71. Murthi, B. P. S. and S. Sarkar. The Role of the Management Sciences in Research on Personalization. *Management Science*, 49(10):1344-1362, 2003.
72. Nakamura, A. and N. Abe. Collaborative filtering using weighted majority prediction algorithms. In *Proc. of the 15th International Conference on Machine Learning*, 1998.
73. Nurnberger, G. *Approximation by Spline Functions.* Springer-Verlag, 1989.
74. Oard, D. W. and J. Kim. Implicit feedback for recommender systems. In *Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98-08*. AAAI Press, 1998.
75. Pavlov, D. and D. Pennock. A Maximum Entropy Approach To Collaborative Filtering in Dynamic, Sparse, High-Dimensional Domains. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems (NIPS'02)*, 2002.
76. Pazzani, M. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, pages 393-408, December 1999.
77. Pazzani, M. and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27:313-331, 1997.
78. Peddy, C. C., and D. Armentrout. *Building Solutions with Microsoft Commerce Server*

2002. Microsoft Press, 2003.
79. Pennock, D. M. and E. Horvitz. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *IJCAI'99 Workshop: Machine Learning for Information Filtering*, August 1999.
80. Popescul, A., L. H. Ungar, D. M. Pennock, and S. Lawrence. Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. In *Proc. of the 17th Conf. on Uncertainty in Artificial Intelligence*, Seattle, WA, 2001.
81. Powell, M. J. D. *Approximation Theory and Methods,* Cambridge University Press, 1981.
82. Ramakrishnan, N., B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy Risks in Recommender Systems. *IEEE Internet Computing*, 5(6):54-62, 2001.
83. Rashid, A. M., I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to Know You: Learning New User Preferences in Recommender Systems. In *Proceedings of the International Conference on Intelligent User Interfaces*, 2002.
84. Robertson S. and S. Walker. Threshold Setting in Adaptive Filtering. *Journal of Documentation*, 56:312-331, 2000.
85. Rocchio, J. J. Relevance Feedback in Information Retrieval. *SMART Retrieval System – Experiments in Automatic Document Processing*, G. Salton ed., PrenticeHall, Ch. 14, 1971.
86. Resnick, P., N. Iakovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 Computer Supported Cooperative Work Conference*, 1994.
87. Rich, E. User Modeling via Stereotypes. *Cognitive Science*, 3(4):329-354, 1979.
88. Rosset, S., E. Neumann, U. Eick, N. Vatnik, and Y. Idan. Customer Lifetime Value Modeling and Its Use for Customer Retention Planning. In *Proc. of the 8th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining (KDD-2002)*, July 2002.
89. Salton, G. *Automatic Text Processing*. Addison-Wesley, 1989.
90. Sarwar B., G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems – a case study. In *Proc. of the ACM WebKDD Workshop*, 2000.
91. Sarwar, B., G. Karypis, J. Konstan, and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. In *Proc. of the 10th International WWW Conference,* 2001.
92. Schaback, R. and H. Wendland. Characterization and construction of radial basis functions. In *Multivariate Approximation and Applications*. Eds. N. Dyn, D. Leviatan, D. Levin and A. Pinkus. Cambridge University Press, 2001.
93. Schafer, J. B., J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1/2):115-153, 2001.
94. Schein, A. I., A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proc. of the 25th Annual Intl. ACM SIGIR Conf.*, 2002.
95. Schmittlein, D. C., D. G. Morrison, and R. Colombo. Counting Your Customers: Who are they and what will they do next? *Management Science*, Vol. 33(1), 1987.
96. Shani, G., R. Brafman, and D. Heckerman. An MDP-based recommender system. In *Proc. of Eighteenth Conference on Uncertainty in Artificial Intelligence*, August 2002.
97. Shardanand, U. and P. Maes. Social information filtering: Algorithms for automating 'word of mouth'. In *Proc. of the Conf. on Human Factors in Computing Systems*, 1995.
98. Sheth, B. and Maes P. Evolving agents for personalized information filtering. In *Proceedings of the 9th IEEE Conference on Artificial Intelligence for Applications*, 1993.
99. Si, L. and R. Jin. Flexible Mixture Model for Collaborative Filtering. In *Proceedings of the 20th International Conference on Machine Learning*, Washington, D.C., August 2003.
100. Soboroff, I. and C. Nicholas. Combining content and collaboration in text filtering. In

IJCAI'99 Workshop: Machine Learning for Information Filtering, August 1999.
101. Somlo, G. and A. Howe. Adaptive Lightweight Text Filtering. In *Proceedings of the 4th International Symposium on Intelligent Data Analysis*, Lisbon, Portugal, September 2001.
102. Statnikov, R. B. and J. B. Matusov. *Multicriteria Optimization and Engineering*. Chapman & Hall, 1995.
103. Terveen, L., W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3):59-62, 1997.
104. Tran, T. and R. Cohen. Hybrid Recommender Systems for Electronic Commerce. In *Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04*, AAAI Press, 2000.
105. Ungar, L. H., and D. P. Foster. Clustering methods for collaborative filtering. In *Recommender Systems. Papers from 1998 Workshop. Technical Report WS-98-08*. AAAI Press, 1998.
106. Wade, W. A grocery cart that holds bread, butter and preferences. *NY Times*, Jan. 16, 2003.
107. Yang, Y. and B. Padmanabhan. On Evaluating Online Personalization, in *Proceedings of the Workshop on Information Technology and Systems*, pp. 35-41, December 2001.
108. Yu, K., X. Xu, J. Tao, M. Ester, and H.-P. Kriegel. Instance Selection Techniques for Memory-Based Collaborative Filtering. In *Proceedings of Second SIAM International Conference on Data Mining (SDM'02)*, 2002.
109. Yu, K., A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel. Probabilistic Memory-Based Collaborative Filtering. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):56-69, 2004.
110. Zaïane, O. R., J. Srivastava, M. Spiliopoulou, B. M. Masand (eds.). *WEBKDD 2002 – Mining Web Data for Discovering Usage Patterns and Profiles* (Lecture Notes in Computer Science 2703), Springer, 2003.
111. Zhang Y. and J. Callan. Maximum Likelihood Estimation for Filtering Thresholds. In *Proc. of the 24th Annual International ACM SIGIR Conference*, New Orleans, LA, 2001.
112. Zhang, Y, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th Annual International ACM SIGIR Conference*, pp. 81-88, 2002.

Gediminas Adomavicius received the PhD degree in computer science from New York University in 2002. He is an assistant professor in the Department of Information and Decision Sciences at the Carlson School of Management, University of Minnesota. Dr. Adomavicius' research focuses on personalization technologies, data mining, and combinatorial auction mechanisms. He has published more than 20 refereed journal and conference papers in these areas. He is a member of the ACM, IEEE, and IEEE Computer Society.

Alexander Tuzhilin received Ph.D. in Computer Science from the Courant Institute of Mathematical Sciences, NYU. He is currently an Associate Professor of Information Systems at the Stern School of Business, NYU. His current research interests include knowledge discovery in databases, personalization and CRM technologies. He published widely in leading CS and IS journals and conference proceedings and served on program committees of numerous CS and IS conferences. Dr. Tuzhilin was as a Co-Chair of the Third IEEE International Conference on Data Mining in 2003. He currently serves on the Editorial Boards of the IEEE Transactions on Knowledge and Data Engineering, the Data Mining and Knowledge Discovery Journal, the INFORMS Journal on Computing, and the Electronic Commerce Research Journal.